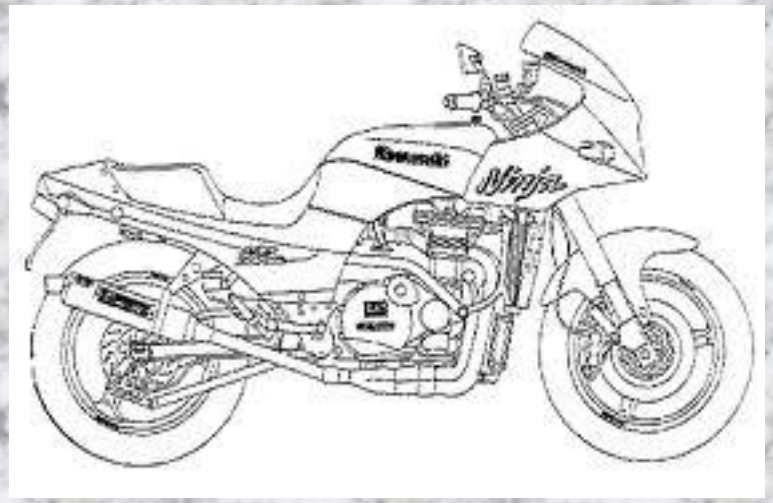
CS 302: Introduction to Programming in Java

Lecture 15



Class

Instances of the class (objects) – only valid at runtime

Private Instance Methods

- Instance methods usually public – why?
- If we have an internal function that we do not want others to be able to call, make it private
- Ex. a sort method on a phonebook object
  - Phonebook could have public methods to add and remove people from the phonebook but should keep itself sorted no matter what
  - Both the add() and remove() instance methods could call a private sort() method

Accessing Instance Variables

- Global scope within the class – why?
- this.varName ALWAYS refers to the instance variable
- varName will refer to the instance variable if the varName is unique

Accessing Instance Variables Example

```
public class BankAccount
{
  private int balance;
  public BankAccount(int balance)
  {
   //what goes here?
  }
}
```

Constructors

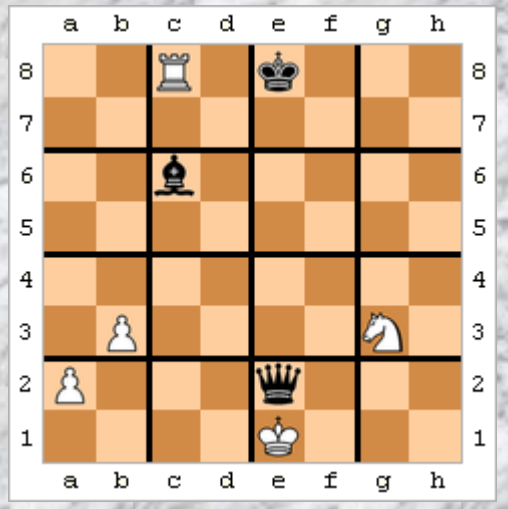
- Form: public <ClassName> (<param list>)
- Can have multiple constructors as long as they take in different parameters
  - The appropriate constructor will be called based on what arguments are passed in
- **Method Overloading: having multiple methods with the same name that take in different arguments**

Practice 1

- Create the class for a Motorcycle Object
- Motorcycles have:
  - Color
  - Current Speed
  - Number of Gears
- Motorcycles can:
  - Accelerate
  - Decelerate

Practice 2

- Create the class for a Chess Piece:
- Chess Pieces have:
  - Color
  - Type (pawn, rook, knight, bishop, queen, king)
  - A grid location (a1 – h8)
  - Status (live or dead)
- Chess Pieces can:
  - Move to a new location
  - Die



Aggregation

- "has a" relationship
- Objects of one class contain objects of another class
- Ex.
  - class Phonebook aggregates the class Contacts
  - class Wizard aggregates the class Wand
  - Mission aggregates Commodity, Location, and String

Object References

- All object variables are reference variables
- Variable stores the memory location of the object, NOT the object itself (think arrays)
- 2 or more object variables can point to the same object
  - Ex: BankAccount dansAccount = suesAccount;
  - Now modifying 1 will modify both
  - Different from: int x = y; modifying x will not change y

Special Instance Methods: toString()

- toString()
  - If I have an object variable and print it out, what happens?
  - To fix this, create a toString() method that returns a String – this will automatically be called if you try to print out your object (System.out.println(varName))
  - Each object will have its own unique String representation (up to you)
  - Ex. A BankAccount might return a String with the balance, the account number, and the date it was created
  - Method header: public String toString() - no params

Special Instance Methods: equals()

- If I have two objects and compare them using ==, what is being compared?
- equals(Object o) is another method commonly implemented to fix this
- Code that goes inside will be unique for all objects
- Ex. for a phonebook, the .equals() method might return true if one phonebook has all the same contacts as another
- Header: public boolean equals(Object o)

Review – Writing Object Classes

- Object classes have 3 parts: Instance data, Constructor(s), and Instance methods
  - Instance Data:
    - Should be declared private and represents the internal data that the object will work with
    - Instance data can be other objects (ex. a quiz could be comprised of question objects)
    - this.variableName will always refer to the instance data for the particular object
  - Constructors
    - Used to create a new instance of this object type
    - Can have multiple as long as they have different parametes
    - Written as: public ObjectName(<params>)
    - Purpose: initialize instance data
  - Instance Methods:
    - Define the way to interact with an object of this type
    - Can be public or private
    - 2 types: Accessors and Mutators

Practice 1

- Create a Contact class
- Contacts have:
  - Name
  - Phonenumber
  - (Address)
- Contacts have accessors and mutators for all their instance data

Practice 2

- Create a Phonebook class
- Phonebooks have:
  - An ArrayList of Contacts
- Phonebooks can:
  - Add/remove contacts
  - Look up the phonenumber of a contact
  - Change the name / phonenumber of a contact
  - Print all contacts names and numbers in alphabetical order